Detección y Explotación de Vulnerabilidades XSS Almacenado y CSRF en DVWA

Objetivo del Ejercicio

El objetivo de este ejercicio fue simular un escenario en el que un atacante explota vulnerabilidades XSS almacenado y CSRF en la aplicación vulnerable DVWA. Se identificó, explotó de forma ética y se propusieron soluciones para ambos fallos de seguridad.

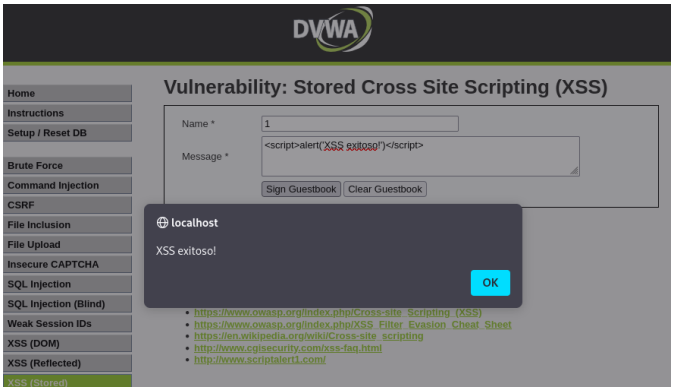
Hallazgos y Evidencia

1. Vulnerabilidad XSS Almacenado

- **Payload utilizado**: Se inyectó el siguiente código JavaScript en el campo de comentarios del módulo "XSS (Stored)".

HTML

<script>alert('XSS exitoso!')</script>



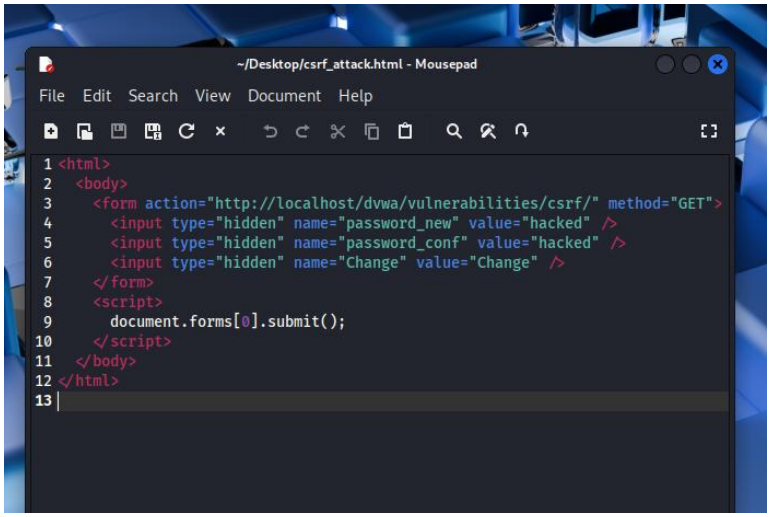
- **Resultado**: Al enviar el formulario, el script se almacenó en la base de datos. Cada vez que la página se carga, el navegador del usuario ejecuta el script, lo que se evidenció con una ventana emergente que mostraba el mensaje "XSS exitoso!". Esto demuestra que el servidor no valida ni sanitiza la entrada del usuario.

2. Vulnerabilidad CSRF

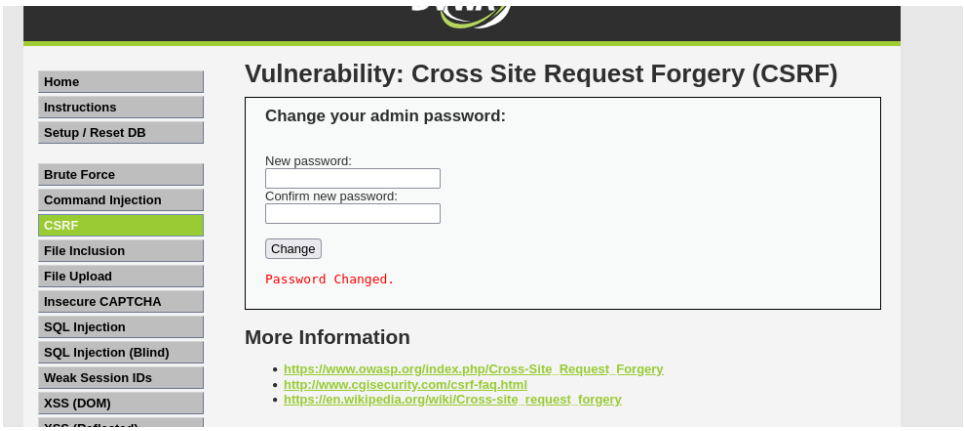
- **Payload utilizado**: Se creó un archivo HTML local (csrf_attack.html) que contenía un formulario oculto. Este formulario realizaba una solicitud GET al endpoint de cambio de contraseña de DVWA.

HTML

```
<html>
 <body>
  <form action="http://localhost/dvwa/vulnerabilities/csrf/" method="GET">
   <input type="hidden" name="password_new" value="hacked" />
   <input type="hidden" name="password_conf" value="hacked" />
   <input type="hidden" name="Change" value="Change" />
  </form>
  <script>
   document.forms[0].submit();
  </script>
 </body>
</html>
```



- **Resultado**: Al abrir el archivo HTML en un navegador con una sesión activa de DVWA, el formulario se envió automáticamente. La contraseña del usuario se cambió con éxito a "hacked" sin que el usuario lo supiera, confirmando la vulnerabilidad.



Análisis de Impacto y Riesgo

- **XSS Almacenado**:
    - **Impacto**: Alto. Un atacante podría robar cookies de sesión, secuestrar cuentas, inyectar formularios de phishing o manipular el contenido de la página para todos los usuarios.
    - **Riesgo**: Alto. La vulnerabilidad es persistente y su impacto se extiende a todos los visitantes de la página.
- **CSRF**:
    - **Impacto**: Crítico. Un atacante podría forzar a un usuario con sesión activa a realizar acciones sensibles (cambios de contraseña, transferencias bancarias, etc.) sin su conocimiento.
    - **Riesgo**: Crítico. La explotación es sencilla y puede comprometer la integridad y confidencialidad de la cuenta de un usuario.

Recomendaciones de Mitigación

Para XSS Almacenado

1. **Sanitización y Validación de Entrada**: Filtra y valida rigurosamente la entrada del usuario en el lado del servidor, permitiendo solo caracteres y formatos seguros.
2. **Escapado de Salida**: Antes de mostrar cualquier contenido en la página, escapa los caracteres especiales (<, >) para que el navegador los interprete como texto y no como código ejecutable.
3. **Content Security Policy (CSP)**: Implementa una política de seguridad de contenido para restringir las fuentes de scripts, evitando la ejecución de código malicioso.

Para CSRF

1. **Tokens Anti-CSRF**: Implementa un token único y aleatorio en cada formulario sensible. Este token debe ser validado en el servidor para asegurar que la solicitud proviene de una sesión legítima y no de un sitio externo.
2. **Validación de SameSite Cookies**: Configura las cookies con el atributo SameSite para evitar que el navegador las envíe en solicitudes de dominios externos.
3. **Cabeceras Referer y Origin**: Valida las cabeceras Referer y Origin en el lado del servidor para verificar que las solicitudes provienen de tu propio dominio.

Preguntas de Reflexión

- **¿Qué diferencia hay entre un XSS reflejado y uno almacenado en cuanto a alcance y persistencia?**
    - **XSS reflejado**: El payload se inyecta y se refleja inmediatamente en la respuesta HTTP. No se almacena en el servidor, por lo que su alcance es limitado al usuario que hizo la petición. Es decir, se necesitan varios ataques para comprometer a más de un usuario.
    - **XSS almacenado**: El payload se guarda en la base de datos y se muestra a todos los usuarios que visiten la página afectada. Es **persistente** y tiene un **alcance masivo**, ya que puede comprometer a múltiples usuarios de forma pasiva.
- **¿Por qué CSRF es especialmente peligroso en usuarios con sesión activa?**
    - El ataque CSRF funciona explotando la confianza que un sitio web tiene en un usuario con sesión activa. La solicitud maliciosa utiliza las cookies de sesión válidas que el navegador envía automáticamente, haciendo que la solicitud parezca legítima. Si el usuario no tiene sesión activa, el ataque no funcionará.
- **¿Cómo se relaciona la validación de origen y los tokens anti-CSRF con la defensa?**
    - Ambos mecanismos se centran en verificar la autenticidad del origen de la solicitud. La **validación de origen** (a través de las cabeceras Referer y Origin) asegura que la solicitud provenga del dominio esperado. Los **tokens anti-CSRF** son una defensa más robusta y eficaz, ya que un atacante no puede adivinar el token único y aleatorio que el servidor espera, evitando que se falsifique la solicitud.